

Artificial Neural Network Based Energy Storage System Modeling for Hybrid Electric Vehicles

Sanjay R. Bhatikar and Roop L. Mahajan

Mechanical Engineering Dept.
University of Colorado at Boulder

Keith Wipke and Valerie Johnson

National Renewable Energy Lab (NREL)

Copyright © 2000 Society of Automotive Engineers, Inc.

ABSTRACT

The modeling of the energy storage system (ESS) of a Hybrid Electric Vehicle (HEV) poses a considerable challenge. The problem is not amenable to physical modeling without simplifying assumptions that compromise the accuracy of such models. An alternative is to build conventional empirical models. Such models, however, are time-consuming to build and are data-intensive. In this paper, we demonstrate the application of an **artificial neural network** (ANN) to modeling the ESS. The model maps the system's state-of-charge (SOC) and the vehicle's power requirement to the bus voltage and current. We show that ANN models can accurately capture the complex, non-linear correlations accurately. Further, we propose and deploy our new technique, **Smart Select**, for designing ANN training data. The underlying philosophy of Smart Select is to design the training data set such that it is uniformly distributed over the entire range of an appropriate ANN output variable, which is typically the variable that is most difficult to model. In this case, we selected training data that were uniformly distributed over the current range. We show that smart-select is economical in comparison with conventional techniques for selection of training data. Using this technique and our in-house ANN software (the CUANN[®]), we developed an artificial neural network model (inputs=2, hidden neurons=3, outputs=2) utilizing only 1583 of the available 32,254 points. When validated on the remaining points, its predictive accuracy, measured by R-squared error, was 0.9978. Next, we describe the integration of the ANN model of the ESS into the MATLAB-SIMULINK environment of NREL's vehicle simulation software, ADVISOR. This yields a simpler implementation of the ESS module in ADVISOR and does away with certain tenuous assumptions in the original implementation.

Lastly, we also show how a dramatic reduction in the size of the training data set may be obtained by applying the **model modifier approach** developed by our research group at the University of Colorado at Boulder.

INTRODUCTION

A growing dependence on foreign oil, along with a heightened concern over the environmental impact of personal transportation, has led the US government to investigate and sponsor research into advanced transportation concepts. One of these future technologies is the hybrid electric vehicle (HEV), typically featuring both an internal combustion engine and an electric motor, with the goal of producing lower emissions while obtaining superior fuel economy. Figure 1 lists the typical components found in an HEV.

The Department of Energy's National Renewable Energy Laboratory (NREL) has developed a HEV simulator, called the ADvanced VehIcle SimulatOR (ADVISOR). This simulator facilitates the optimization of HEV configurations with different subsystems, for best fuel economy and emission level. ADVISOR requires models of the individual components of a HEV such as the propulsion unit and the energy storage unit.

One way to develop models of HEV components is through rigorous analytical procedure. This is typically time-consuming and the simplifying assumptions required to make the analysis tractable impair the value of such models. An alternative is to employ conventional empirical methods, which are variations of the classical regression theme. These models are unwieldy and are usually only suitable for low-end non-linearities. In this paper, we present artificial neural network modeling as a practical alternative to analytical and empirical methods that is accurate and easy to use.

THE ARTIFICIAL NEURAL NETWORK

An Artificial Neural Networks (ANN) is a massively parallel, highly interconnected system of computational nodes or neurons (Figure 3). The neurons are organized into layers. Typically, there is an input layer into which the input vector (independent variables) is fed in, an output layer that produces the output vector (dependent variables), and one or more layers in between. A typical ANN is shown in Figure 3. It has an input layer, a hidden layer and an output layer. There are connections going from a neuron in a layer to all neurons in the next layer. There are no connections between neurons of the same layer. Each connection is associated with a weight. Such an ANN can be trained to learn an input-output mapping by showing it examples of the mapping. During the training procedure, the weights of the ANN are adjusted such that the ANN converges to the output of the training data.

Consider the processing performed by an ANN. An input vector is presented at the input layer and run through the ANN to obtain the output vector. The output of any layer becomes the input to the following layer. The following equation relates the output of a layer to its input from the preceding layer and the interconnection weights between the two layers:

$$Y_i^l = F\left(\sum_{j=1}^{N_{l-1}} w_{ij}^l Y_j^{l-1} + b_i^l\right)$$

Here, Y_i^l is the output of the i th neuron in the l th layer, w_{ij}^l is the weight of the connection from the j th neuron in the $(l-1)$ th layer to the i th neuron in the l th layer, b_i^l is the bias connected to the i th neuron in the l th layer and N_{l-1} is the number of neurons in the $(l-1)$ th layer. F is the activation function, which may be thought of as providing a non-linear gain for the artificial neuron. It is typically a sigmoid function (Figure 4) and bounds the output from any neuron in the network.

$$F = \frac{1}{(1 + e^{-u})}$$

To train an ANN for developing an input-output mapping, data are required which are representative of the mapping. The first step of training is the forward pass, which consists of calculating the output vector by running the input vector through the ANN. This is followed by a backward pass where the error derivatives are calculated for each weight. The error derivatives for a weight are summed until all the data points have been run through the network once. This constitutes an epoch. The weights are updated after each epoch such that the ANN error decreases. Refer to references 2 and 3 for further details.

For this project, we used our in-house ANN software, the 'Colorado University Artificial Neural Network' (CUANN®).

THE ADVANCED VEHICLE SIMULATOR (ADVISOR)

ADVISOR is software created in MATLAB® for the simulation of hybrid electric vehicles (HEVs). ADVISOR allows an HEV to be configured from a selection of components, including several energy storage units, internal combustion engines and drivetrain systems. A graphic user interface, as shown in Figure 5, allows the user to select individual components and configure the HEV.

The simulation routines are implemented in the SIMULINK® toolkit of MATLAB®. SIMULINK® is a graphical programming language. Figure 6 shows the implementation of a series HEV in SIMULINK®.

ADVISOR simulates a **drive cycle**, which consists of a temporal required-speed profile. A drive-cycle constitutes the basis of a simulation. From the operational characteristics of the various HEV components, the vehicle configured by the user attempts to meet the speed profile of the selected drive cycle. Various performance parameters such as emissions, fuel consumption and fuel efficiency are computed.

In this paper, we demonstrate the utility of an artificial neural network as a fast and accurate modeling tool for HEV components. As the performance of ADVISOR depends upon the quality of the component models, the artificial neural network as a modeling tool fills a critical need. Moreover, the ANN can be seamlessly integrated with the ADVISOR system.

THE ENERGY STORAGE SYSTEM (ESS)

The ESS is the source of electrical power in an HEV. It comprises a bank of batteries. Batteries store and deliver electrical energy chemically by initiating and reversing chemical reactions respectively. Although a battery is a simple electrical energy storage device that delivers and accepts energy, the highly non-linear nature of its electrochemical processes makes it difficult to model. It is therefore an excellent candidate for validating the ANN as a modeling tool.

Figure 7 represents the general scheme of the ESS model as required in ADVISOR. The ESS accepts a power request (P_r), and depending on the state-of-charge (SOC) of the battery pack, returns the bus voltage (V) and current (I). The ESS output power is simply the product of the bus voltage and current.

Artificial Neural Network models were developed to implement this scheme.

ARTIFICIAL NEURAL NETWORK MODELING

Experimental data for development of ANN models were collected from a lead-acid battery, with an 'ABC-150' experimental test rig (See Figure 8). Five different drive cycles were simulated by means of this test-rig. Data

from these five drive cycles were used to train ANNs. The complete data set comprised 32,254 examples.

PERFORMANCE MEASURES

In order to gauge the performance of an ANN model on a data set, two measures of error were adopted. These are the average mean squared error (MSE) and the correlation coefficient, R^2 . They are defined as follows

$$MSE = \frac{1}{N} \sum_{i=1}^N (t_i - y_i)^2$$

$$R^2 = 1 - \frac{\sum_{i=1}^N (t_i - y_i)^2}{\sum_{i=1}^N (t_i - t_m)^2}$$

Here,

N is the total number of data points,

t_i is the target output,

t_m is the mean of the target output values, and

y_i is the model's predicted output.

Both these measures have drawbacks and are not good under all situations. However, referring to them both gives a good idea of the ANN model's performance.

ARTIFICIAL NEURAL NETWORK MODEL – I

ANNs were developed with training data sampled at random from the available data. This is a simple and unsophisticated approach. The training data set comprised 6000 points (1200 from each of five driving cycles). ANNs were trained and then validated on the entire data set of 32,254 points.

Figures 9 and 10 represent the validation performance of the ANN model. Figure 9 is a plot of the target voltage versus the ANN's prediction. The MSE is close to zero and the R-squared error is nearly 1. Figure 10 is a plot of the target current versus the ANN's prediction. The MSE is 39.52 and the R-squared error is 0.9850. It was observed that the performance dropped sharply for extreme values of current. This behavior was characteristic of all ANN models developed with data derived by random sampling from the drive cycles.

It was reasoned that this discrepancy arose due to the uneven distribution of training data with respect to current. This is clearly evident from Figure 11, which is a histogram showing distribution of the experimental data vis-a-vis current. More than 70% of the data are confined to less than 30% of the current space.

It was therefore decided to sample training data for an even distribution over the current space. This forms the basis of our **smart select** technique, described in the next section.

SMART SELECT

The underlying philosophy of the Smart Select technique is to design training data set such that the data are uniformly distributed over the entire range of an appropriate output variable. This variable is the one that is most difficult to model by the ANN. Conventional Design of Experiments (DOE) mandates an even distribution of training data over input (independent) variables. Such a full-factorial DOE scheme is data-intensive. It is subject to the 'curse of dimensionality' whereby the size of the training data set increases exponentially with the resolution of the independent variables. A partial-factorial DOE attenuates the problem of data-explosion. However, even with partial factorial DOE, the time required to select training data from an available data set is prohibitively large. This problem is what Smart Select solves. By applying Smart Select, the data are sampled so that the training data have an even distribution vis-à-vis one output (dependent) variable only. The variable selected is the output variable that is more intractable than the others.

For the ANN model of the ESS, the intractable output variable was the current. Accordingly, the training data set was designed with the objective of ensuring an even distribution of data with respect to the current.

ARTIFICIAL NEURAL NETWORK MODEL - II

ANNs were developed with data selected by the Smart Select technique. Smart Select was implemented by a C++ program (see reference 1). 1583 data points were sampled from the driving cycles. The points were sampled in approximately equal measure from each driving cycle. Refer to Figure 12. Clearly, the training data are evenly distributed with respect to current.

An ANN model was developed with these 1583 points. The best ANN configuration comprised 1 hidden layer with 4 neurons. Figure 13 shows the validation performance of the ANN for current. The MSE is 5.78 and the R-squared error is 0.9978. This ANN model met NREL's performance targets: R-squared error ≥ 0.99 and MSE less than that of the original algorithm (MSE < 17.872)

INTEGRATION OF ANN INTO ADVISOR

The ANN described in the previous section was incorporated into ADVISOR. ANN models were created in MATLAB®, using the Neural Network Toolbox. A back-propagation neural network is created in MATLAB® using the "net" command. The SIMULINK® block diagram of an ANN is created using the "gensim" command.

Pilot tests demonstrated the superior learning capability of the CUANN® as compared to ANNs implemented in MATLAB®. Therefore, ANN models were developed using the CUANN® software and the weights were

transferred to identical ANNs created in MATLAB®. See Appendix A for details.

MODEL MODIFIER TECHNIQUE

An important aspect of this investigation was to test the efficacy of our model modifier approach for economy of training data. This approach consists of developing a physical ANN model trained on a first-principle physical model, and then updating this model by a 'shot-of-reality' from experimental data. The physical ANN model is trained with data generated by simulation of the physical model for combinations of input parameters dictated by a statistical design of experiments (DOE). Its accuracy is limited as the physical model is impaired by simplifying assumptions. By introducing experimental data a shot-of-reality is delivered which compensates for this impairment. At the same time, fewer experimental data are required as the physical ANN model captures the essential physical underpinnings of the model. The implementation of this concept comprises two stages. The first stage is the physical ANN model. The second stage is the model modifier ANN that introduces a shot-of-reality. Two alternative modes for conjunction of these stages are described in the following section.

IMPLEMENTATIONS OF THE MODEL MODIFIER APPROACH

The two alternative implementations of the model modifier approach (Marwah and Mahajan, 1999) are:

A. The Difference Method – This implementation is represented schematically in Figure 14. The model modifier ANN is trained to predict the difference between the output of the physical ANN model and the experimental data. The expectation is that, if the difference is a simpler function than the target function, fewer data points will be required to build an accurate model, as compared to building the model from scratch.

B. The Source-Input Method – This method is represented schematically in Figure 15. In this method, the output of the physical ANN constitutes an extra input to the model modifier ANN. Since the physical ANN model is close to the target model, its output serving as input to the model modifier simplifies the learning task of the latter. This reduces the number of experimental data points required for developing the model. For example, in the degenerate case when the physical ANN is identical to the target model, the learning problem of the model modifier ANN is reduced to auto-association.

APPLICATION OF THE MODEL MODIFIER APPROACH TO THE ESS.

The physical ANN model was developed from a partly empirical, partly analytical model of the ESS. This model is represented in Figure 16. Stage-I is an empirical stage that computes the open-circuit voltage (V_{oc}) and the internal resistance (R_{int}) of the battery from the SOC, by means of statistical regression models (described subsequently). These variables, along with the vehicle's

power requirement constitute the inputs to Stage-II, which computes the bus voltage (V) and current (I). The empirical correlations of Stage-I were developed from a statistical regression analysis of data collected from tests performed on a lead-acid battery whereby the battery was charged/discharged at a uniform rate. The source data for these regression models is presented in Table I. The empirical correlations of Stage-I are as follows:

- A. SOC- V_{oc} correlation -

$$V_{oc} = 11.69531 + 1.520591 \times (\text{SOC}) - 0.409091 \times (\text{SOC})^2 + 0.073815 \times (\text{SOC})^3$$
- B. SOC- R_{int} (charge) correlation [piecewise regression model] -
 1. $R_{int}(\text{charge}) \mid \text{SOC} \in (0, 0.6)$

$$R_{int} = 0.031544 - 0.018697 \times (\text{SOC}) + 0.033902 \times (\text{SOC})^2 + 0.02702 \times (\text{SOC})^3 - 0.117424 \times (\text{SOC})^4$$
 2. $R_{int}(\text{charge}) \mid \text{SOC} \in (0.6, 1)$

$$R_{int} = 1.3347 - 7.388749 \times (\text{SOC}) + 15.45125 \times (\text{SOC})^2 - 14.225 \times (\text{SOC})^3 + 4.875 \times (\text{SOC})^4$$
- C. SOC- R_{int} (discharge) correlation -

$$R_{int} = 0.048467 - 0.093393 \times (\text{SOC}) + 0.057976 \times (\text{SOC})^2$$

The analytical correlations of Stage-II, which result from elementary circuit law analysis, are -

$$I = \frac{V_{oc} - \sqrt{V_{oc}^2 - 4R_{int}P_r}}{2R_{int}}$$

$$V = V_{oc} - IR_{int}$$

Training data for the physical ANN model were generated by simulation using this two-stage physical model. The experimental data comprised a pool of 29,424 points representing a single drive cycle simulation.

RESULTS

Table II shows the results of applying the model modifier approach. The results are represented graphically in Figure 17. The physical ANN model, by itself, had an accuracy of nearly 84%. The model modifier ANN improved the performance considerably. Even a tiny shot of reality consisting of only 6 points improved the accuracy by over 10% (R^2 -error). Further, as observed in Figure 17, the improvement in performance saturated at a shot-of-reality comprising 20 data points. To develop a conventional ANN model of comparable accuracy, 4047 experimental data points were required.

RESULTS AND CONCLUSION

We have demonstrated that the ESS of an HEV can be adequately modeled by an artificial neural network. We have also demonstrated the effectiveness of the Smart-

Select technique for designing the training data for an ANN. The performance of the best ANN trained by random sampling of data was:

- Mean Squared Error: 39.52
- R-Squared Error: 0.9850

By application of the Smart-Select technique for selection of training data, the performance was improved to:

- Mean Squared Error: 5.78
- R-Squared Error: 0.9978

The ANN's performance was better than that of the original, circuit-law analysis based algorithm, which was:

- Mean Squared Error: 17.872
- R-Squared Error: 0.9653

Figure 18 shows the results screen of ADVISOR with the ANN successfully incorporated. The topmost graph of this figure shows the profiles of the required speed versus the attained speed over the drive cycle. Time is plotted on the X-axis. As the required speed is met throughout the drive cycle, the two plots coincide. The next graph shows the SOC history over the drive cycle. The last two graphs show the vehicle's power request and the corresponding actual power achieved over the drive cycle.

Further, we have demonstrated the effectiveness of the model modifier approach. By applying this approach, a saving of 96% was achieved in the amount of experimental data required for training.

ACKNOWLEDGMENTS

We are grateful to the U.S. Department of Energy and the National Energy Renewable Lab (NREL) for sponsorship of this investigation. We are thankful to the University of Colorado, under whose auspices this research was carried out. The authors gratefully acknowledge the contribution of Dr. Yuan Li, Matthew Cuddy and David Rausen to this project.

REFERENCES

1. Bhatikar, S. R., *Artificial Neural Network Base Energy Storage System Modeling for Hybrid Electric Vehicles*, MS Thesis, University of Colorado at Boulder, 1999.
2. Haykin, S., *Neural Networks: A Comprehensive Foundation*. New York, MacMillan College Publishing Company, 1994.
3. Marwah, M., Mahajan, R. L., *Building Neural Network Equipment Models Using Model Modifier Techniques*. IEEE Trans. Semiconductor Mfg., Vol. 12, No. 3, (August), pp. 377-381, 1999.
4. Marwah, M., Li Y., Mahajan, R. L., *Integrated Neural Network Modeling For Electronic Manufacturing*. J. Electronics Manufacturing, Vol. 6, No. 2 (June), pp. 79-91, 1996.
5. Marwah, M., *Neural Network Modeling Techniques For Electronics Manufacturing Processes*. Masters' Thesis. University of Colorado at Boulder, 1993.
6. Smith, M., *Neural Networks for Statistical Modeling*. New York. Van Nostrand Reinhold, 1993.

For more about HEVs and ADVISOR:

7. <http://www.ccts.nrel.gov>

DEFINITIONS, ACRONYMS, ABBREVIATIONS

ANN – Artificial Neural Network.

ADVISOR – Advanced Vehicle Simulator.

CUANN – Colorado University Artificial Neural Network.

DOE – Design of Experiments.

ESS – Energy Storage System.

HEV – Hybrid Electric Vehicle.

MSE – Mean Squared Error.

APPENDIX A.

Porting between CUANN® and MATLAB®.

For this project, the CU-ANN® was required to be ported into the MATLAB® environment of ADVISOR. The porting is explained here with the help of a simple example. To port an ANN developed by the CUANN® software into the MATLAB® environment, it is first required to define an ANN of identical structure in MATLAB®, with its Neural Network Toolbox. To define a feed-forward neural network in MATLAB®, the Neural Network Toolbox provides the command `newff`. The anatomy of this command is described with the following example -

```
net = newff([0 1; 0 1], [3, 2], {'logsig', 'logsig'});
```

Ranges of the
input variables,
in order.

Sizes of
the
layers, in
order.

Activation
functions
of the
layers, in
order.

In this example, the variable `net` is a two-layer, feed-forward neural network. That means there is one hidden layer, the other layer being the output layer. The hidden layer has three neurons and there are two output neurons. The activation function of both layers is a logistic sigmoid. Note that the range of each input variable is specified as `[0 1]`, since the CU-ANN® automatic pre-processor normalizes the input variables in that range.

The next step is to load the weights of the ANN from the CU-ANN®. For this purpose, text files are created from the CU-ANN® as follows:

- A text file for the bias weights of each layer.
- A text file for the interconnection weights of each layer.

According to the format of the weight files in the CU-ANN®, the weights are organized layer-wise, as matrices. The bias weights of a layer and the weights of interconnections with its previous layer are organized as a matrix. The weights of an individual neuron are organized in a row, with the first element in a row being its bias weight. The rows are arranged in the order of the neurons. In the example cited, the weights of the hidden layer would be represented as follows –

	Bias	Input 1	Input 2	Input 3
Neuron 1
Neuron 2
Neuron 3

The text file for loading the bias weights of a layer would comprise the first column of its weight matrix in the CU-ANN weight file. The text file for loading its interconnections with its previous layer would comprise the remaining columns. In the example cited, the weight assignment would require four text files, two for each layer. The weight assignment in MATLAB® would be as follows:

```
net.IW{1, 1}=weights_10;  
net.b{1, 1}=weights_10b;  
net.LW{2, 1}=weights_21;  
net.b{2, 1}=weights_21b;
```

In the MATLAB® environment, the hidden layer connecting to the input layer is identified as `net.IW{1, 1}`. The weights of any other layer are identified as `net.LW{a, b}` where `a` is the number of the layer and `b` is the number of the layer with which it is interconnected. The numbering system starts with the first hidden layer in the feed-forward direction, which is standard practice. The same notation has been used in this example to label the text files, with a trailing 'b' indicating that the file contains bias weights. Now, the ANN is ready for prediction. The Neural Network Toolbox provides the command `sim` for this purpose. The anatomy of this command is as follows:

```
a = sim(net, in)
```

output:
vectors
in row
format

network
variable

input:
vectors in
row format

Note that the input vectors are presented to the network in row format. Correspondingly, the outputs are also in row format. The inputs require to be normalized in the range [0, 1]. The normalization is performed as follows:

$$in^i(normalized) = \frac{in^i - in_{\min}^i}{in_{\max}^i - in_{\min}^i}$$

where the superscript labels the input variable, and the maximum and minimum values refer to the training data. The CU-ANN normalizes the output variables in the range [0.2, 0.8], so the output a has to be de-normalized as follows:

$$a^j(de-normalized) = \frac{a^j - 0.2}{0.8 - 0.2} \times (a_{\max}^j - a_{\min}^j)$$

where, the superscript labels the output variable, and the maximum and minimum values refer to the training data.

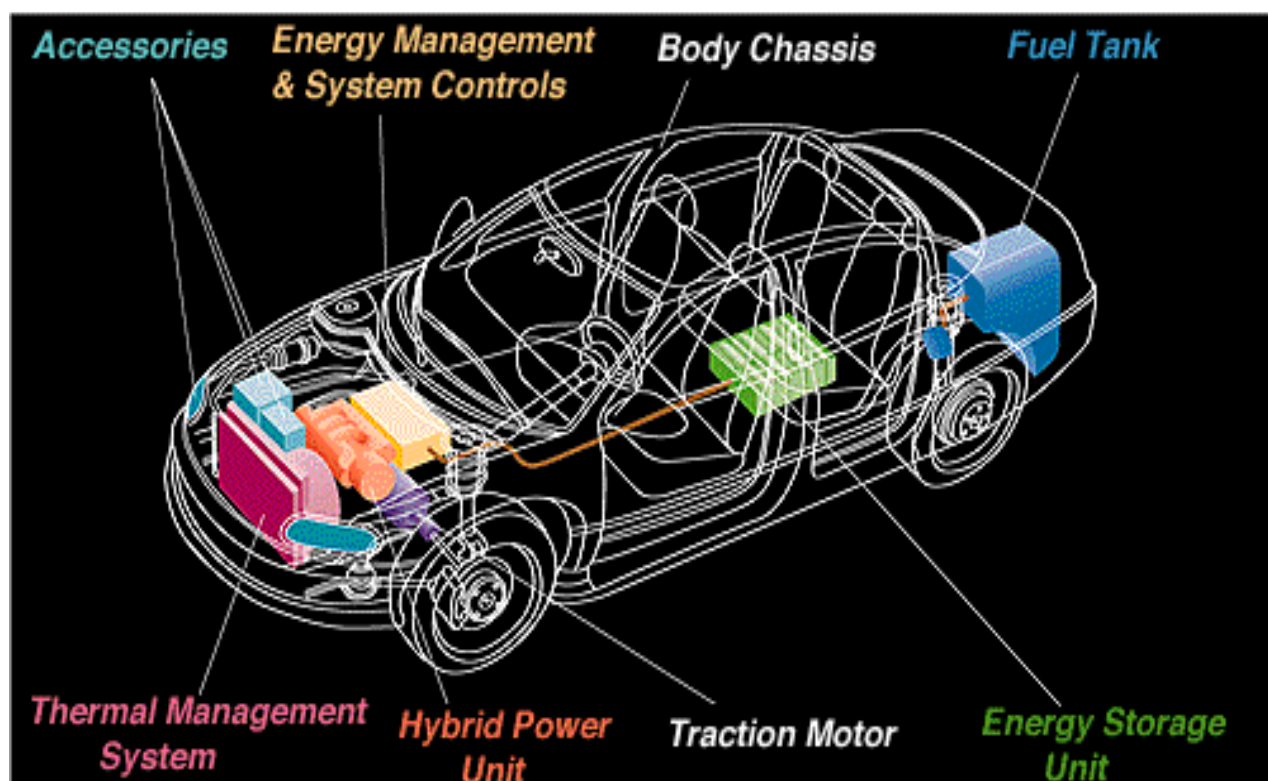


Figure 1: Typical HEV subsystems.

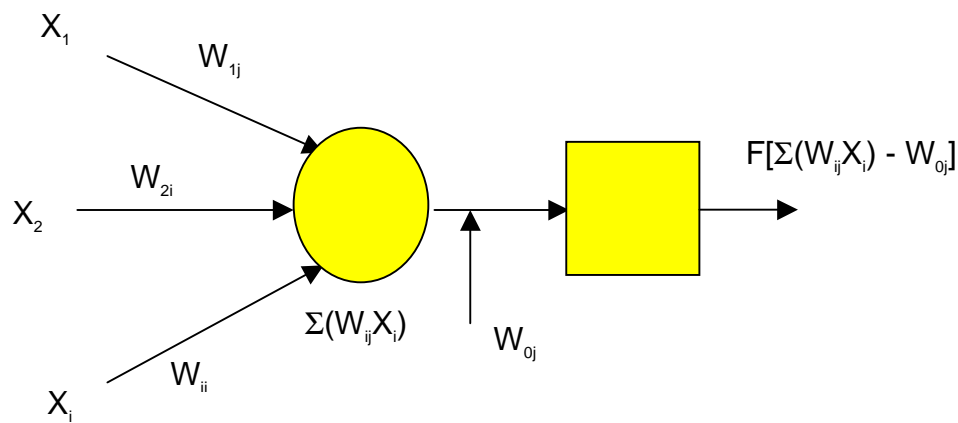


Figure 2: The artificial neuron.

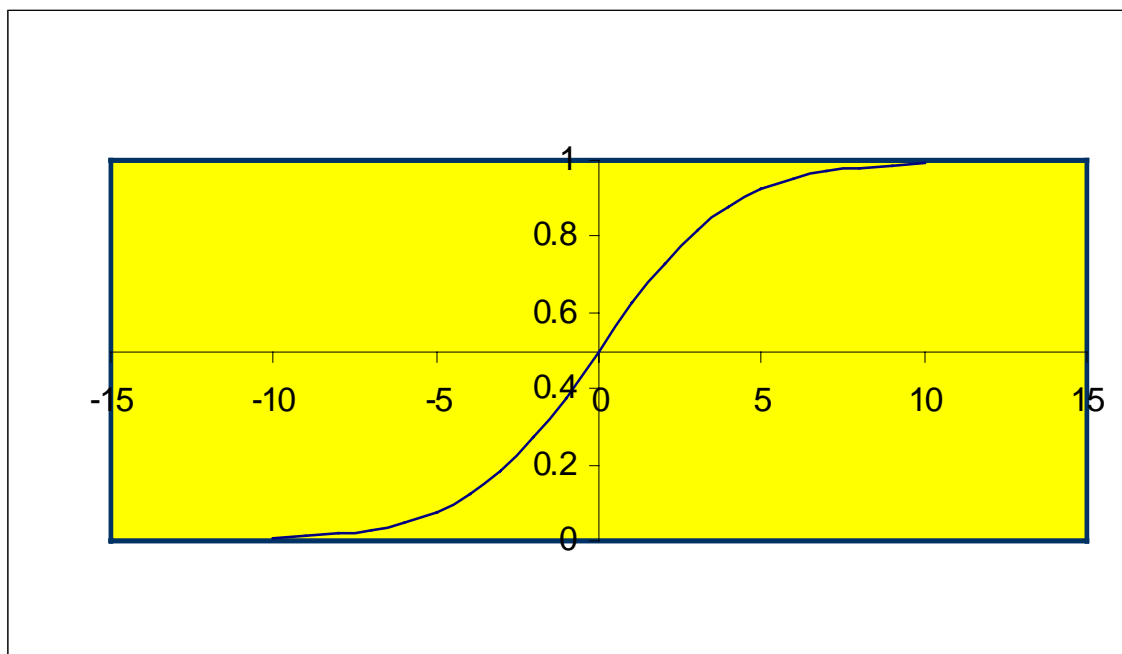


Figure 4: The sigmoid activation function.

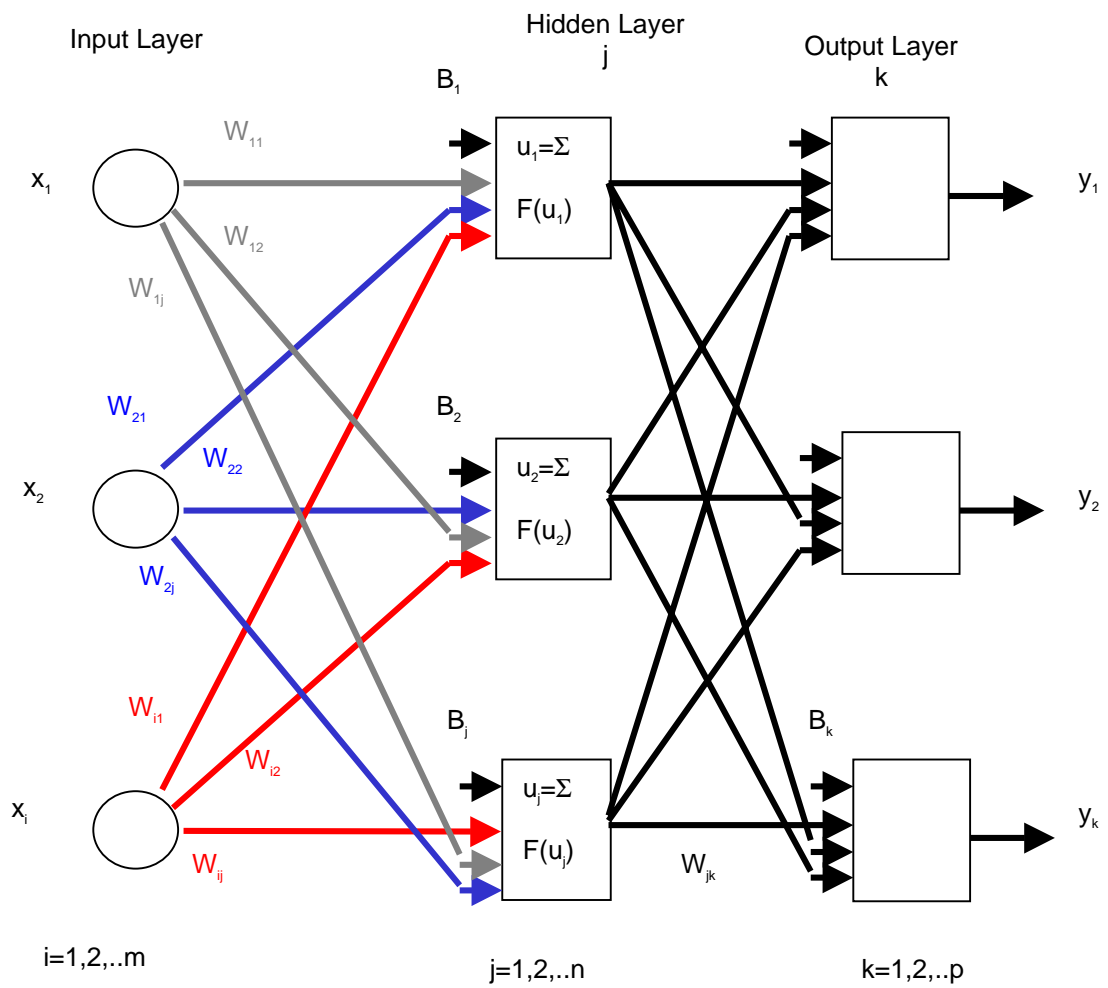


Figure 3: An artificial neural network.

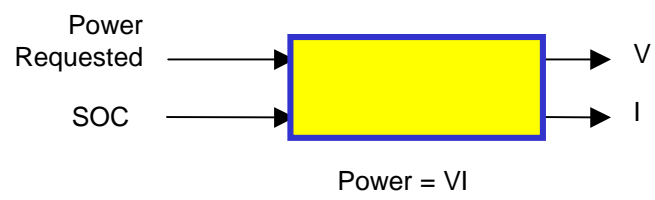


Figure 7: A schematic of the ESS.



Figure 8: ABC-150 experimental test rig.

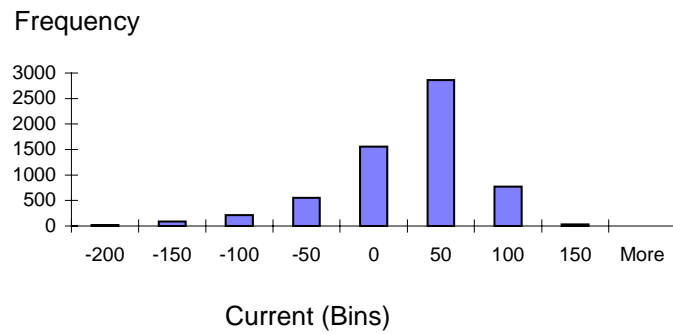


Figure 11: Uneven distribution of data vis-à-vis current.

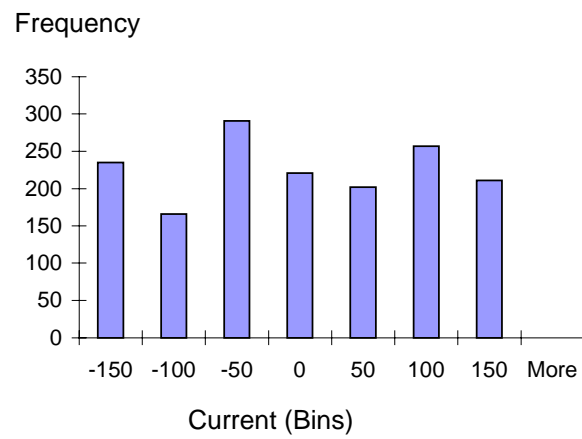


Figure 12: Even distribution of training data vis-à-vis current with Smart-Select.

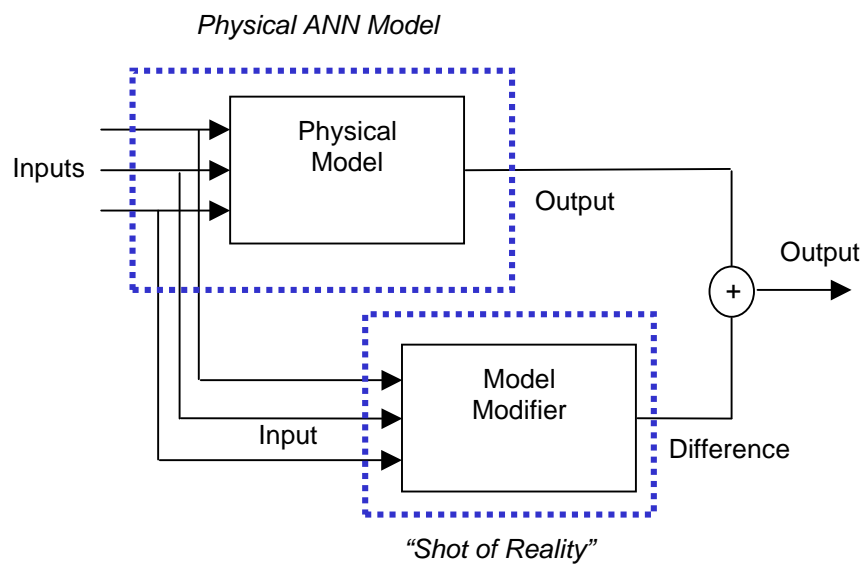


Figure 14: The 'difference' implementation of the model modifier approach.

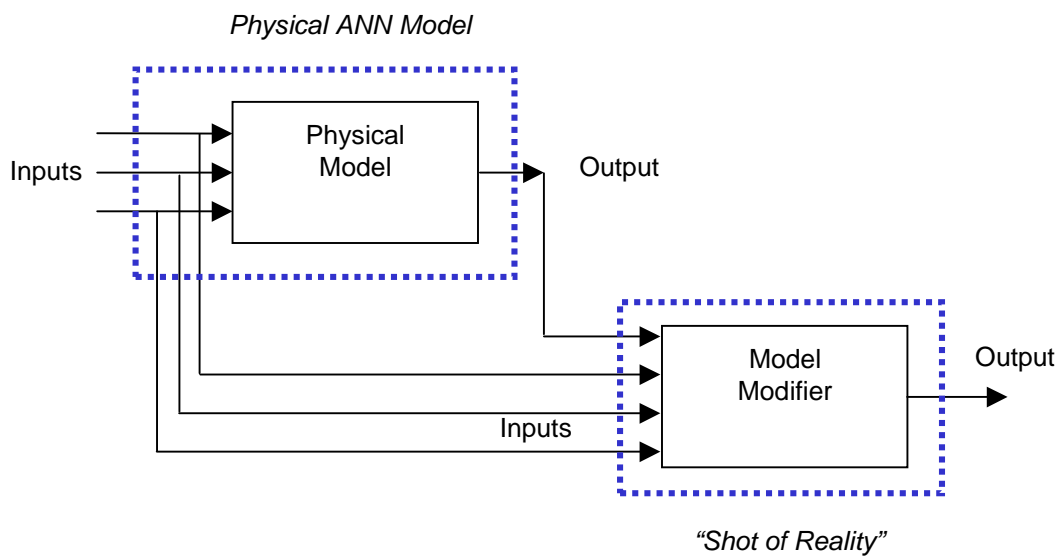


Figure 15: The 'output-as-input' implementation of the model modifier approach.

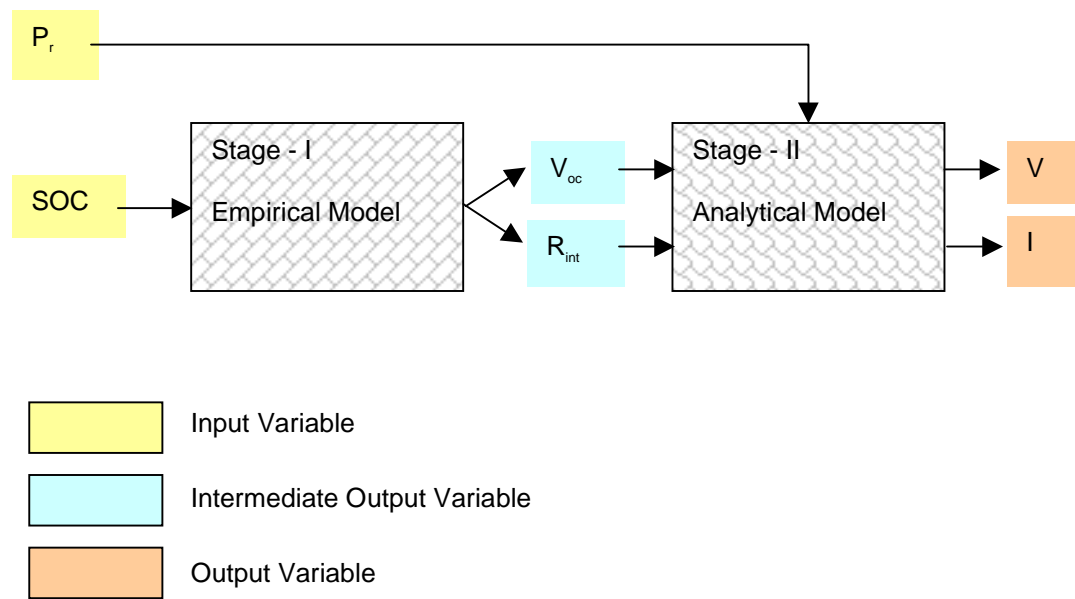


Figure 16: Physical model for the model modifier approach as applied to the ESS.

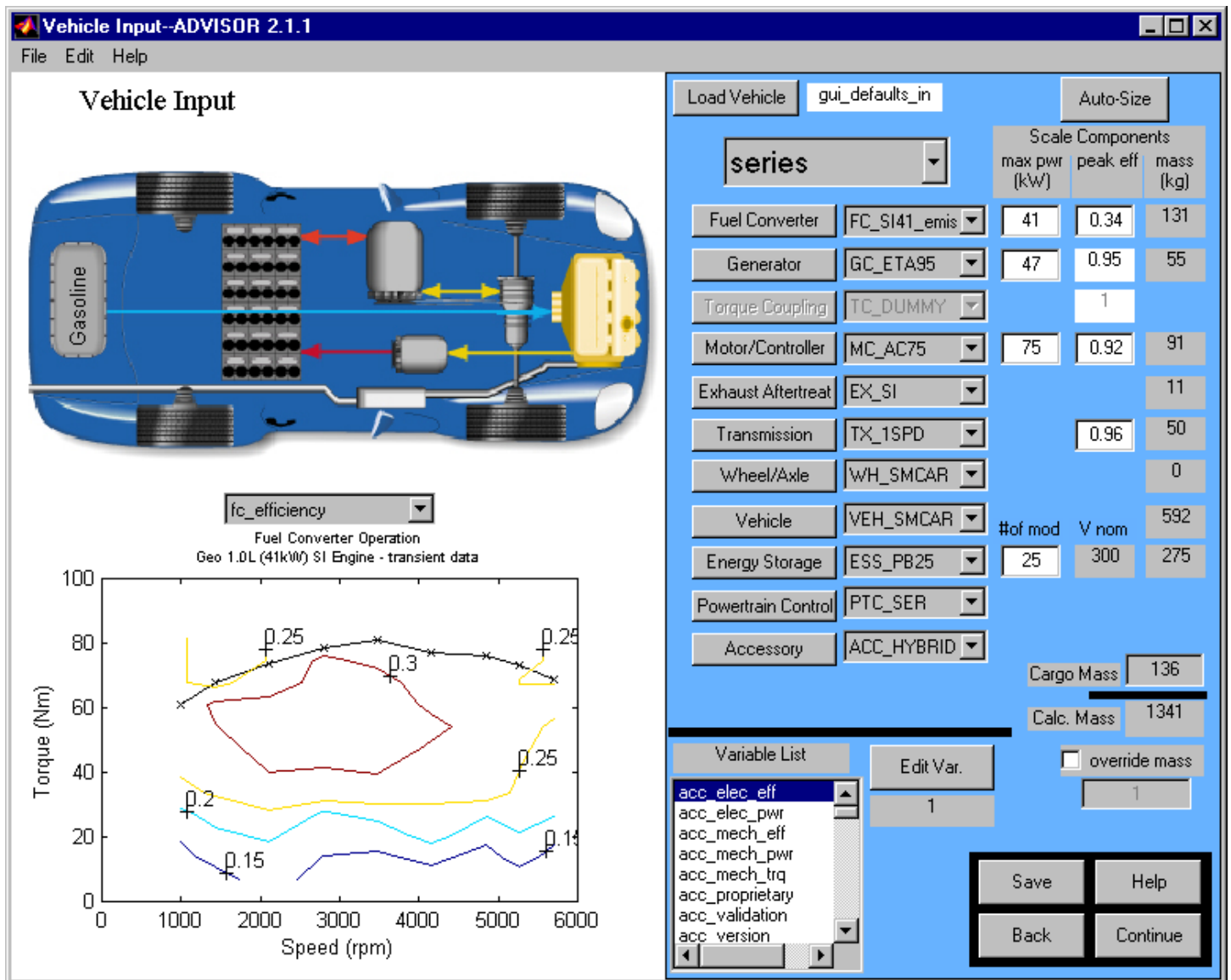


Figure 5: Graphic user interface for ADVISOR.

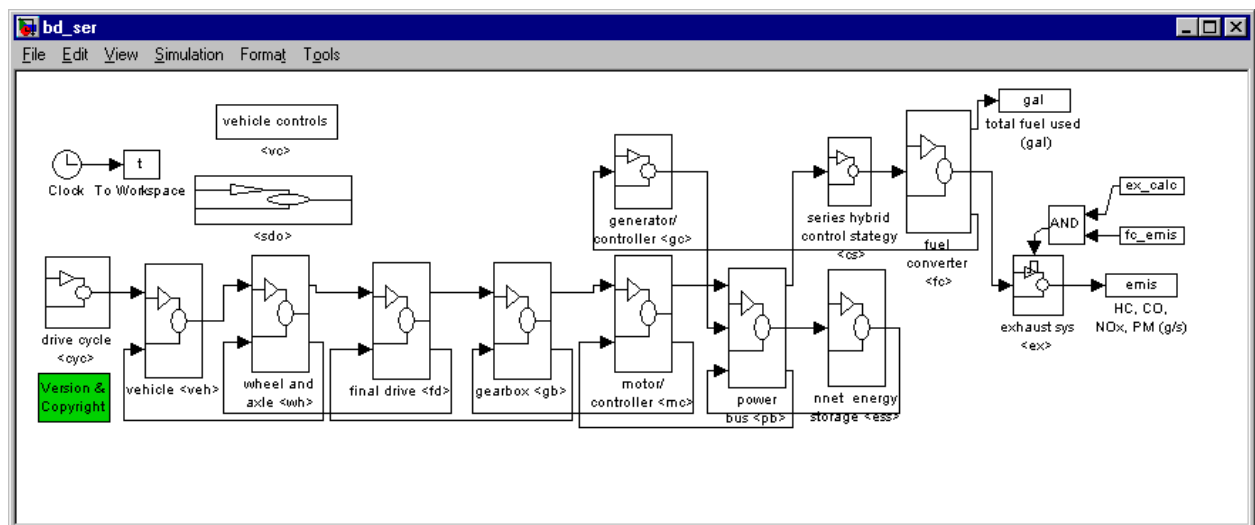


Figure 6: SIMULINK implementation of a series HEV in ADVISOR.

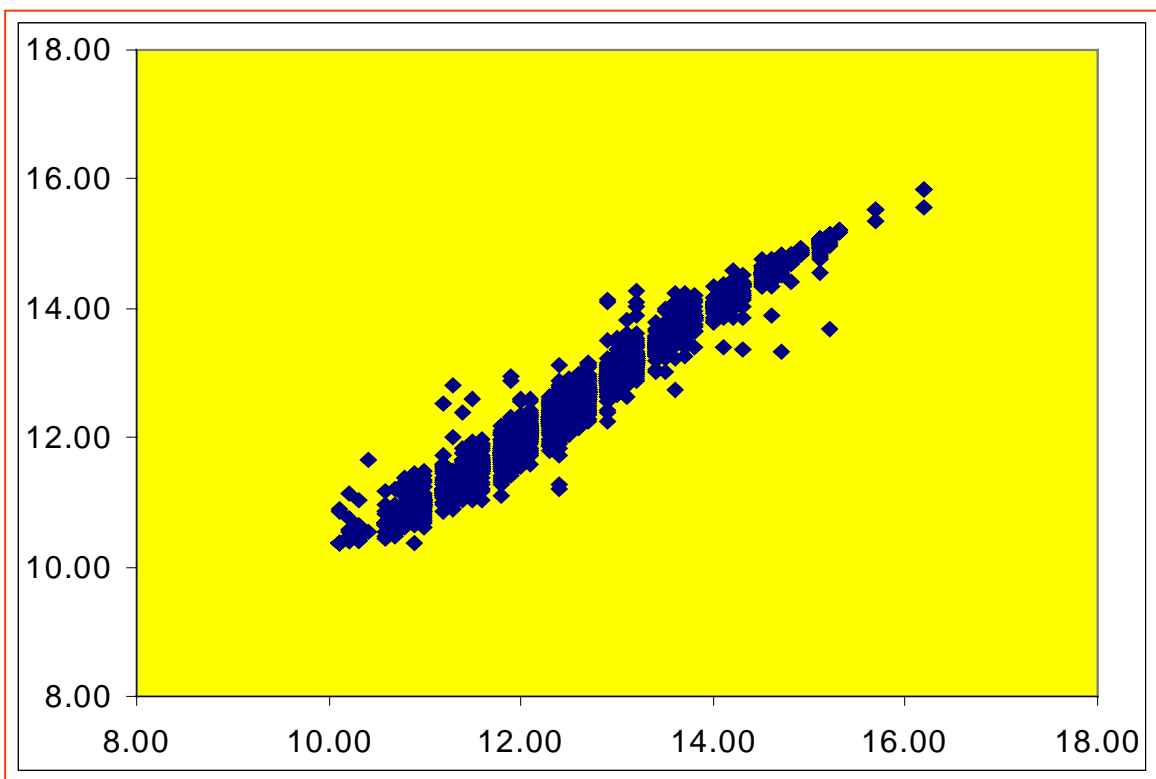


Figure 9: ANN Validation (Voltage)

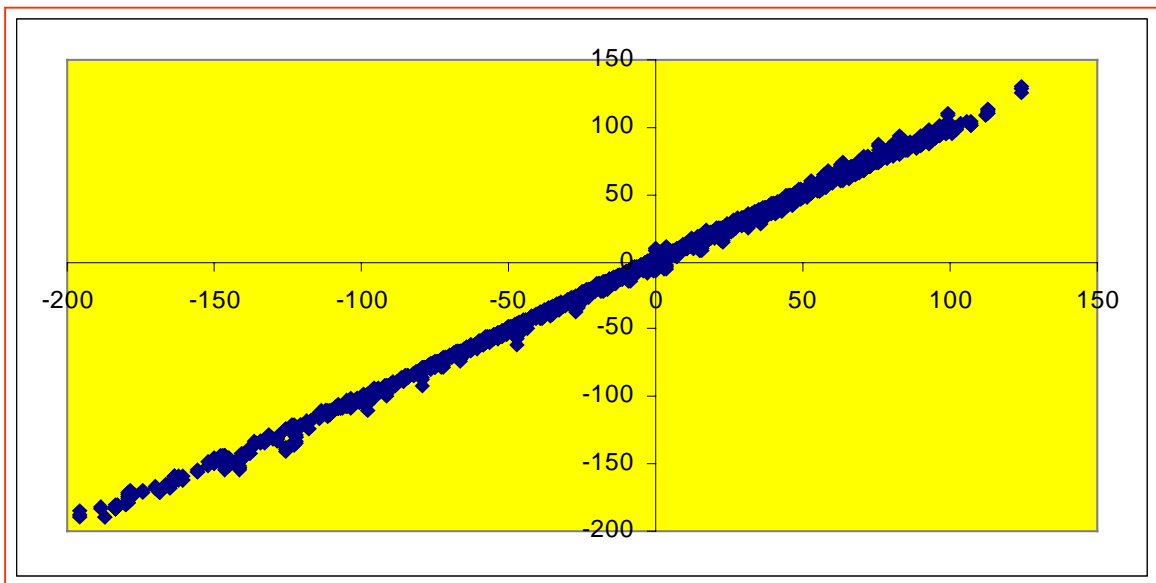


Figure 13: ANN Validation (Current).

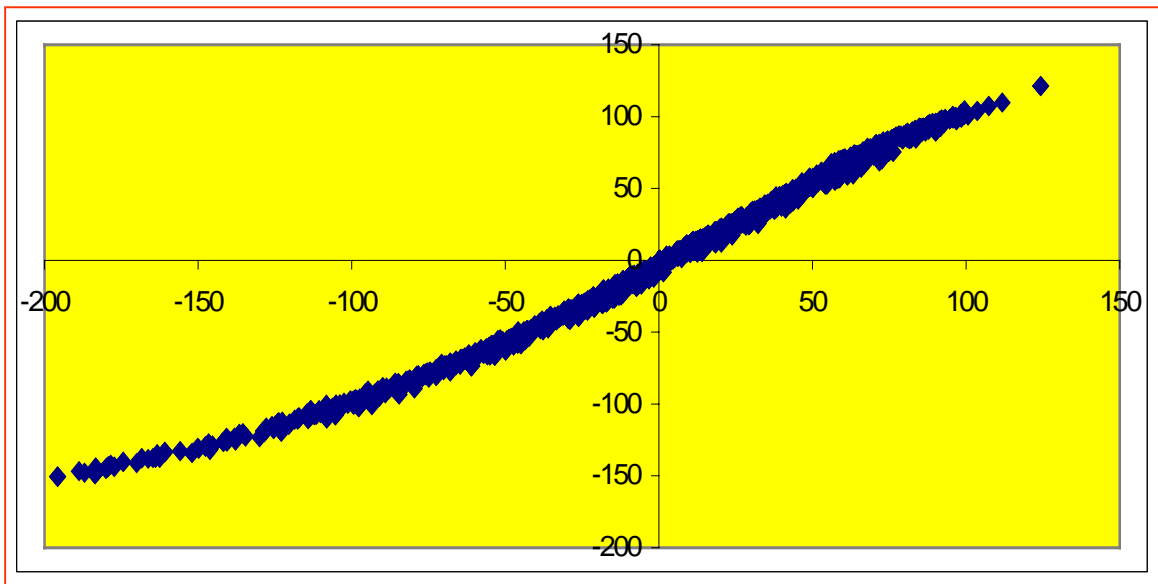


Figure 10: ANN Validation (Current).

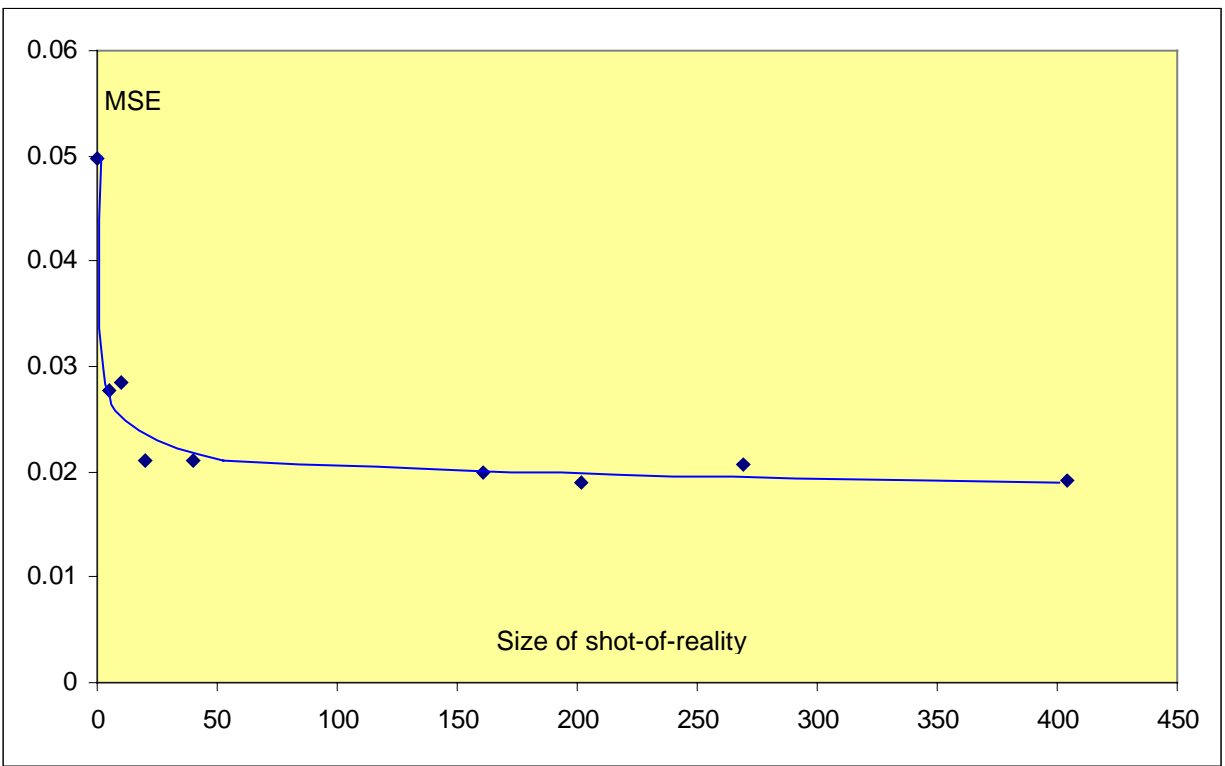


Figure 17: Results of application of the model modifier approach to the ESS.

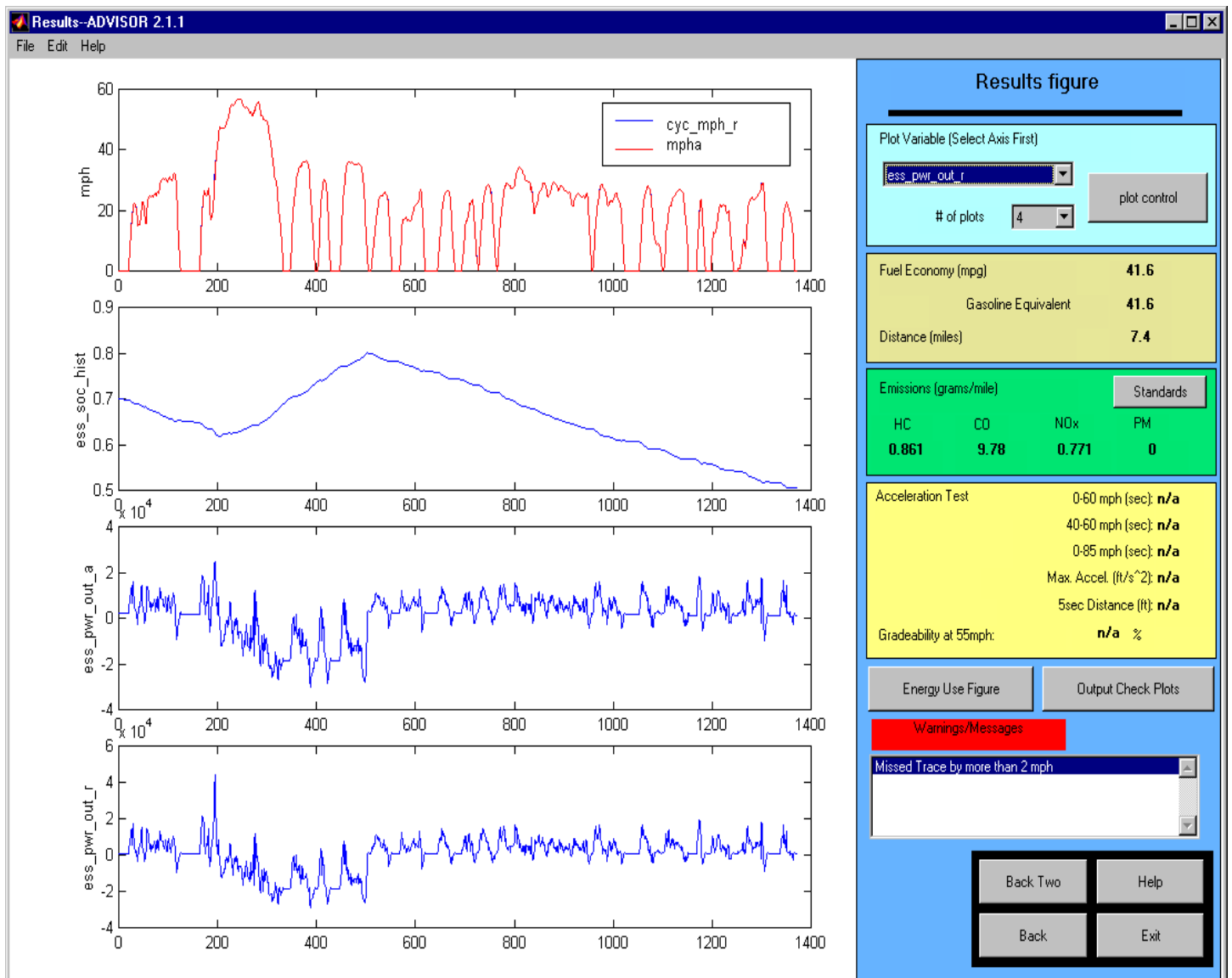


Figure 18: Successful integration of ANN in advisor.

Table I: Source data for empirical part of the physical ESS model.

SOC	V_{oc}	R_{int}	
		Charge	Discharge
0	11.70	0.0316	0.0407
0.1	11.85	0.0298	0.0370
0.2	11.96	0.0295	0.0338
0.3	12.11	0.0287	0.0269
0.4	12.26	0.028	0.0193
0.5	12.37	0.0269	0.0151
0.6	12.48	0.0231	0.0131
0.7	12.59	0.025	0.0123
0.8	12.67	0.0261	0.0117
0.9	12.78	0.0288	0.0118
1	12.89	0.0472	0.0122

Table II: Results of the model modifier approach as applied to the ESS.

Table II: Model Modifier Approach		
Shot of Reality (%)	MSE	R-squared Error (%)
0	0.04973	0.837555
10	0.01911	0.979956
6.6	0.02072	0.97559
5	0.01905	0.979369
4	0.01992	0.978514
1	0.02100	0.976343
0.5	0.02104	0.976997
0.25	0.02856	0.953965
0.125	0.02768	0.956616